Uncoded Caching and Cross-level Coded Delivery for Non-uniform File Popularity

Emre Ozfatura and Deniz Gündüz
Department of Electrical and Electronic Engineering
Imperial College London
Email: {m.ozfatura, d.gunduz}@imperial.ac.uk

***Abstract*—Proactive content caching at user devices and coded delivery is studied considering a non-uniform file popularity distribution. A novel centralized uncoded caching and coded delivery scheme, which can be applied to large file libraries, is proposed. The proposed *cross-level coded delivery (CLCD)* scheme is shown to achieve a lower average delivery rate than the state of art. In the proposed CLCD scheme, the same sub-packetization is used for all the files in the library in order to prevent additional zero-padding in the delivery phase, and unlike the existing schemes in the literature, two users requesting files from different popularity groups can be served by the same multicast message in order to reduce the delivery rate. Simulation results indicate significant reduction in the average delivery rate for typical Zipf distribution parameter values.**

I. Introduction

Caching and coded delivery can improve the network load and latency significantly. This is shown in [1] by considering a server holding N files serving K users, each equipped with its own cache memory of size M files, over a shared error-free link. The proposed solution consists of two phases. In the *placement phase*, files are divided into sub-files and each user stores certain sub-files. In the *delivery phase*, the server multicasts carefully constructed XORed combinations of the requested sub-files, and each user recovers its request from its cache contents and the multicast messages. In the proposed scheme, the server controls the caching decisions of the users. The authors showed in [2] that, under certain assumptions, similar gains can be achieved via coded delivery with *decentralized caching*, where users cache randomly. In [3], the authors introduce a new delivery scheme that utilizes the common requests in order to further reduce the delivery rate. These schemes are built on the assumption that user requests are uniform over the library. However, in on-demand video streaming networks video popularity statistics are not homogeneous [4], [5].

The more interesting case of non-uniform demands has been recently considered in [6]–[11]. The common approach in [6]–[9] is to group the files according to their popularities, distribute the available cache memory among the groups, and then use the coded delivery scheme in [2] to deliver all the missing files. All the aforementioned works try to achieve the optimal placement structure according to file popularities. In particular, it is shown in [9] that the achievable delivery rate with using two groups of files is at most a constant factor away from the lower bound. Although the proposed methods are successful in reducing the delivery rate for non-uniform demands, they are sub-optimal policies in general.

An optimal content placement strategy for a particular coded delivery scheme, the one introduced in [1], has been independently proposed in [10] and [11]. In this scheme, each file is divided into $K+1$ disjoint fragments, and the kth fragment of each file is cached as in the placement phase of [1] with parameter $t = k-1$. The authors show that the optimal sizes of these fragments can be obtained by solving a linear optimization problem. Although this scheme allows each file to be divided into different size fragments, when the file library is large, the proposed scheme tends to divide the library into only a small number of groups, while the sizes of the fragments in each group are identical. This is because [10] and [11] optimize the content placement for the delivery phase in [1], which is, actually, a suboptimal delivery method for multi-level content placement. In particular, we observe that the scheme in [10] tends to classify the files into two groups according to their popularities, where only the popular files are cached, and this is done in an identical manner. Since the less popular files are not cached at all, if one of them is requested, it is delivered via unicast transmission, which increases the delivery rate significantly. This motivates the proposed *cross-level coded delivery (CLCD)* scheme, which codes sub-files from different placement levels. Our proposed scheme first optimizes the coded delivery phase according to the given placement levels, and then optimizes the placement phase accordingly, i.e, the number of files in each cache level. We show that the CLCD scheme provides a significant reduction in the average delivery rate (up to 10%), while also reducing the complexity of the placement and delivery algorithms.

II. System Model

We consider a single content server with a database of N files, each of size F bits, denoted by $W_1, \dots, W_N$, serving K users, each with a cache memory of capacity MF bits. The users are connected to the server through a shared error-free link. We follow the two-phase model in [1]. Caches are filled during the placement phase without the knowledge of particular user demands. User requests are revealed in the delivery phase, and are satisfied simultaneously.

This work was supported in part by the Marie Sklodowska-Curie Action SCAVENGE (grant agreement no. 675891), and by the European Research Council (ERC) Starting Grant BEACON (grant agreement no. 725731).

The request of user k is denoted by d_k, $d_k \in [N] \triangleq \{1,\dots,N\}$, and the demand vector is denoted by $\mathbf{d} \triangleq (d_1,\dots,d_K)$. The corresponding delivery rate $R(\mathbf{d})$ is defined as the total number of bits sent over the shared link, normalized by the file size. We assume that each user requests file W_n with probability p_n, where $p_1 \geq p_2 \geq \cdots \geq p_N$, and $\sum_{n=1}^{N} p_n = 1$. Let $P(\mathbf{d})$ denote the probability of observing demand vector $\mathbf{d}$. We want to minimize the average delivery rate $\bar{R}$, defined as

$$\bar{R} \triangleq \sum_{\mathbf{d}} P(\mathbf{d}) R(\mathbf{d}). \tag{1}$$

A file W_n is said to be cached at *level* t, if it is divided into $\binom{K}{t}$ non-overlapping sub-files of equal size, and each sub-file is cached by a distinct subset of t users. Then, each sub-file can be identified by an index term $\mathcal{I}$, where $\mathcal{I} \subseteq [K]$ and $|\mathcal{I}| = t$, such that sub-file $W_{n,\mathcal{I}}$ is cached by users $k \in \mathcal{I}$. Following a placement phase in which all the files are cached at level t, as proposed in [1], in the delivery phase, for each subset $\mathcal{S} \subseteq [K]$, $|\mathcal{S}| = t+1$, all the requests of the users in $\mathcal{S}$ can be served simultaneously by multicasting

$$\bigoplus_{s \in \mathcal{S}} W_{d_s, \mathcal{S}\setminus\{s\}}. \tag{2}$$

Thus, with a single multicast message the server can deliver $t+1$ sub-files, and achieve a *multicasting gain* of $t+1$.

III. Cross-level Coded Delivery

Here, we explain the proposed CLCD scheme. Initially, the file library is divided into two groups: the most popular N_h files are called the *high-level* files, while the remaining $N-N_h$ files are called the *low-level* files. In the placement phase, each file is divided into $\binom{K}{t}$ sub-files. High-level files are cached at level t; that is, each user caches $\binom{K-1}{t-1}$ sub-files of each high-level file, while the low-level files are cached at level 1; that is, each user exclusively caches $\binom{K-1}{t-1}/t$ sub-files of each low-level file. We denote this particular case of the CLCD scheme by CL(t,1), where t and 1 represent the caching levels of the high and low-level files, respectively. For each low-level file, the set of all sub-files are divided into K disjoint and equal-size subsets[1], so that each user caches the sub-files in a different subset. Hence, for a given N_h and N the cache size required for the CL(t, 1) scheme is given by

$$M(N_h, N, t, 1) \triangleq \frac{N_h(t-1)+N}{K}. \tag{3}$$

For given M, N_h can be obtained from (3). Hence, the grouping of the library is directly related to t, M, and N. Next, we explain the CL(2,1) scheme in detail.

A. CL(2, 1) Scheme

We first present the achievable delivery rate of the CL(2, 1) scheme, and then explain the placement and delivery phases in detail. For a given demand realization $\mathbf{d}$, the users requesting high-level files are called the *high-level users*, and are denoted by $\mathcal{K}^h$. Similarly, users requesting low-level files are called the *low-level users*, and are denoted by $\mathcal{K}^l$. Let $k^a \triangleq |\mathcal{K}^a|, a \in \{l,h\}$. We will show that for a cache size of $M(N_h, N, t, 1)$, and demand realization $\mathbf{d}$, the following delivery rate is achievable by the proposed CL(2, 1) scheme:

$$R(\mathbf{d}) = \begin{cases} \dfrac{\binom{k^h}{3} + \left\lceil \left(L(\mathbf{d}) - 3\binom{k^h}{3} \right)/2 \right\rceil}{\binom{K}{2}}, & \text{if } k^h \geq 2 \\ \dfrac{K-1}{2} & \text{otherwise} \end{cases}, \tag{4}$$

where $L(\mathbf{d})$ is the total number of missing sub-files for demand realization $\mathbf{d}$, and is given by

$$L(\mathbf{d}) \triangleq (K-1)\left[\frac{K^2}{2} - k^h - \frac{k^l}{2}\right]. \tag{5}$$

In the coded delivery scheme of [1], when all the files are cached at level t, the delivery rate is $\frac{K-t}{t+1}$. Similarly in (4), when $k^l = K$, there are only low level users and the delivery rate is $\frac{K-1}{2}$. The delivery rate depends only on $\mathbf{k}_{2-1} \triangleq [k^h, k^l]$; rather than the demand vector; hence, the dependence on $\mathbf{d}$ in (4) and (5) can be replaced by $\mathbf{k}_{2-1}$. Note that k^h and k^l are random variables, and their distribution $P_{N_h,N}(\mathbf{k}_{2-1})$ depends on N_h, N, and the popularity distribution. The average load can now be written as follows:

$$\bar{R} \triangleq \sum_{\mathbf{k}_{2-1}} R(\mathbf{k}_{2-1}) P_{N_h,N}(\mathbf{k}_{2-1}). \tag{6}$$

User	1	2	3	4	5	6	7
index of cached sub-files	{1,2}	{1,2}	{1,3}	{1,4}	{1,5}	{1,6}	{1,7}
	{1,3}	{2,3}	{2,3}	{2,4}	{2,5}	{2,6}	{2,7}
	{1,4}	{2,4}	{3,4}	{3,4}	{3,5}	{3,6}	{3,7}
	{1,5}	{2,5}	{3,5}	{4,5}	{4,5}	{4,6}	{4,7}
	{1,6}	{2,6}	{3,6}	{4,6}	{5,6}	{5,6}	{5,7}
	{1,7}	{2,7}	{3,7}	{4,7}	{5,7}	{6,7}	{6,7}

TABLE I: Sub-files cached by each user in Example 1.

B. Placement phase

In the placement phase, file W_k is divided into $\binom{K}{2}$ sub-files, denoted by $W_{k,\{i,j\}}$, $k \in [N]$, $i,j \in [K]$. If W_k is a high-level file, $W_{k,\{i,j\}}$ is stored by users i and j. Hence, each user stores $K-1$ sub-files for each high-level file. On the other hand, each user exclusively stores only $\frac{K-1}{2}$ sub-files for each of the low-level files.

Example 1. Consider $K = 7$ users and $N = 7$ files. Let $\{A, B, C, D, E, F, G\}$ denote the file library with decreasing popularity, where F and G are low-level, whereas A, B, C, D and E are high-level files. Each file is divided into $\binom{K}{2} = 21$ sub-files, and each user stores $K-1 = 6$ sub-files of each high-level file, and $\frac{K-1}{2} = 3$ sub-files of each low-level file. User cache contents after the placement phase is illustrated in Table I; for each user the sub-files in red are cached for all the files, whereas the sub-files in blue are cached for only

[1]The only requirement is that $\binom{K}{t}$ should be divisible by K. When K is a prime number this requirement holds for any t.

High-level users	Multicast message
1 2 3	$A_{23} \oplus B_{13} \oplus C_{12}$
1 2 4	$A_{24} \oplus B_{14} \oplus D_{12}$
1 2 5	$A_{25} \oplus B_{15} \oplus E_{12}$
1 3 4	$A_{34} \oplus C_{14} \oplus D_{13}$
1 3 5	$A_{35} \oplus C_{13} \oplus E_{13}$
1 4 5	$A_{45} \oplus D_{15} \oplus E_{14}$
2 3 4	$B_{34} \oplus C_{24} \oplus D_{23}$
2 3 5	$B_{35} \oplus C_{25} \oplus E_{23}$
2 4 5	$B_{45} \oplus D_{25} \oplus E_{24}$
3 4 5	$C_{45} \oplus D_{35} \oplus E_{34}$
Low-level users	Multicast message
6 7	$G_{16} \oplus F_{17}$
6 7	$G_{26} \oplus F_{27}$
6 7	$G_{67} \oplus F_{37}$

TABLE II: Multicasted messages in the first two steps of the delivery phase in Example 1.

high-level files.[2]

C. Delivery phase

Delivery phase of the proposed scheme consists of four steps, which will be explained on Example 1.

Example 1 continued. Assume that different high-level files are requested by the first five users, i.e., $\mathcal{K}^h = \{1,2,3,4,5\}$, and different low-level files by users 6 and 7, i.e., $\mathcal{K}^l = \{6,7\}$. The delivery phase is carried out in four steps:

1) Intra-high-level delivery: The first step of the delivery phase is identical to that in (2) for $t = 2$. The only difference is that, now we consider only the users in $\mathcal{K}^h$, instead of $[K]$.

2) Intra-low-level delivery: The second step also follows (2) with $t = 1$, targeting low-level users in $\mathcal{K}^l$. In Example 1, the messages delivered by the server corresponding to the first two steps are listed in Table II.

3) Cross-level delivery: This step is the main novelty of the CLCD scheme. First, note that each high-level user has $(K-1)/2$ sub-files in its cache that are requested by a low-level user. For instance, in Example 1, user 1 has sub-files $\{G_{12}, G_{13}, G_{14}\}$ that are requested by user 7. Let $\mathcal{H}_{i,j}$ denote the set of sub-files stored at high-level user i that are requested by low-level user j, e.g., $\mathcal{H}_{1,7} \triangleq \{G_{12}, G_{13}, G_{14}\}$. Similarly, let $\Gamma_{i,j}$ be the set of $\binom{K-2}{1} = K-2$ sub-files stored by low-level user j, that are requested by high-level user i, e.g., $\Gamma_{1,7} \triangleq \{A_{27}, A_{37}, A_{47}, A_{57}, A_{67}\}$ in Example 1. We note that sub-files $\{A_{27}, A_{37}, A_{47}, A_{57}\}$ are also cached by a high-level user, but sub-file A_{67} is cached by only low-level users. At this point we introduce the set $\Omega_{i,j} \subseteq \Gamma_{i,j}$ of the sub-files that are requested by high-level user i, and cached by low-level user j as well as by a high-level user, e.g., $\Omega_{1,7} \triangleq \{A_{27}, A_{37}, A_{47}, A_{57}\}$. Further, we introduce the set Λ_i, $i \in \mathcal{K}^h$, of the sub-files that are requested by high-level user i and cached by only low-level users, e.g., $\Lambda_1 = \{A_{67}\}$.

In the third step our aim is to deliver all the sub-files requested by the low-level users and the sub-files requested by the high-level users that are cached by only low-level users, via multicast messages, each destined for one high-level and one low-level user. More formally, we want low-level user j, $j \in \mathcal{K}^l$, to recover all the sub-files in $\cup_{i \in \mathcal{K}^h} \mathcal{H}_{i,j}$, and we want high-level user i, $i \in \mathcal{K}^h$, to recover all the sub-files in Λ_i.

Now, we introduce the sets $\mathcal{F}_{i,j} \subseteq \Gamma_{i,j}$ for all $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$, which satisfy the following constraints

$$|\mathcal{H}_{i,j}| = |\mathcal{F}_{i,j}|, \ \forall i \in \mathcal{K}^h, \ \forall j \in \mathcal{K}^l, \tag{7}$$

$$\Lambda_i \subseteq \cup_j \mathcal{F}_{i,j}, \ \forall i \in \mathcal{K}^h, \tag{8}$$

$$\mathcal{F}_{i,j} \cap \mathcal{F}_{i,k} = \emptyset, \ \forall i \in \mathcal{K}^h \text{ and } j,k \in \mathcal{K}^l. \tag{9}$$

Let us first assume that sets $\mathcal{F}_{i,j}$ satisfying (7)-(9) exist for all $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$. From (7), one can observe that it is possible to construct a one-to-one mapping between the elements of $\mathcal{H}_{i,j}$ and $\mathcal{F}_{i,j}$.

The proposed coded delivery procedure for the third step works as follows: first, for each $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$ the sub-files in sets $\mathcal{H}_{i,j}$ and $\mathcal{F}_{i,j}$ are paired via a one-to-one mapping, and the server multicasts the XOR of the paired sub-files. We remark that the sub-files in $\mathcal{H}_{i,j}$ are requested by low-level user j, and are available at high-level user i, while the sub-files in $\mathcal{F}_{i,j}$ are requested by high-level user i, and are available at low-level user j. Hence, with each multicast message both high-level user i and low-level user j receives one sub-file of their respective requests. Since all the sub-files in $\cup_{i \in \mathcal{K}^h, j \in \mathcal{K}^l} \mathcal{H}_{i,j}$ are delivered via multicast messages, the low-level users collect all their missing sub-files. Furthermore, (8) ensures that each high-level user collects its missing sub-files that are available only in the caches of the low-level users. As a final remark, (9) guarantees that high-level users do not receive the same sub-file multiple times.

Next, we show how to construct the sets $\mathcal{F}_{i,j}$, $i \in \mathcal{K}^h$, $j \in \mathcal{K}^l$. In order to ensure (8) and (9), Λ_i is partitioned into subsets $\{\Lambda_{i,j}\}_{j \in K^l}$ with approximately uniform cardinality, i.e., $|\Lambda_{i,k}| - |\Lambda_{i,j}| \leq 1$, $\forall j,k \in \mathcal{K}^l$, $j \neq k$, and such that $\Lambda_{i,j} \subseteq \Gamma_{i,j}$ holds for all $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$. Further details of this approximately uniform partitioning are provided in [12]. We note that, if $\Lambda_{i,j} \subseteq \mathcal{F}_{i,j}$, then (8) holds. We also assume that the same partitioning is applied to all Λ_i's. Partitions of Λ_i's for Example 1 are illustrated in Table III. For given $\Omega_{i,j}$ and $\Lambda_{i,j}$, $\mathcal{F}_{i,j}$ can be constructed as follows; $\mathcal{F}_{i,j} = \Lambda_{i,j} \cup \{\Omega_{i,j} \setminus \Delta_{i,j}\}$, for some $\Delta_{i,j} \subseteq \Omega_{i,j}$. From the construction, one can easily verify that (9) holds. Finally, in order to ensure (7), we need to show that it is possible to construct $\Delta_{i,j}$, $i \in \mathcal{K}^h$, $j \in \mathcal{K}^l$, which satisfy the following equality

$$|\mathcal{H}_{i,j}| = |\Lambda_{i,j}| + |\Omega_{i,j}| - |\Delta_{i,j}|. \tag{10}$$

We note that, if the following inequality holds

$$|\Lambda_{i,j}| \leq |\mathcal{H}_{i,j}| \leq |\Lambda_{i,j}| + |\Omega_{i,j}|, \tag{11}$$

then, $\Delta_{i,j}$ satisfying (10) can be always found.

From the construction, we know that $|\mathcal{H}_{i,j}| = \frac{K-1}{2}$; however, $|\Omega_{i,j}|$ and $|\Lambda_{i,j}|$ depend on the realization of the user demands, i.e., $|\Omega_{i,j}| = k^h - 1$ and $|\Lambda_{i,j}| = \left\lceil \frac{k^l - 1}{2} \right\rceil$ or

[2] In Example 1, we use a systematic placement for the low-level files; that is, sub-file $W_{k,\{i,j\}}$ is cached by either user i or j. This systematic placement enables dynamic cache replacement when a low-level (high-level) file becomes a high-level (low-level) file.

(i,j)	$\Omega_{i,j}$	$\Lambda_{i,j}$	$\Delta_{i,j}$	$\mathcal{F}_{i,j}$	$\mathcal{H}_{i,j}$	Multicasted messages
(1,6)	$\{A_{26}, A_{36}, A_{46}, A_{56}\}$	$\emptyset$	$\{A_{46}\}$	$\{A_{26}, A_{36}, A_{56}\}$	$\{F_{12}, F_{13}, F_{14}\}$	$A_{26} \oplus F_{12}$, $A_{36} \oplus F_{13}$, $A_{56} \oplus F_{14}$
(1,7)	$\{A_{27}, A_{37}, A_{47}, A_{57}\}$	$\{A_{67}\}$	$\{A_{27}, A_{47}\}$	$\{A_{67}, A_{37}, A_{57}\}$	$\{G_{12}, G_{13}, G_{14}\}$	$A_{67} \oplus G_{12}$, $A_{37} \oplus G_{13}$, $A_{57} \oplus G_{14}$
(2,6)	$\{B_{16}, B_{36}, B_{46}, B_{56}\}$	$\emptyset$	$\{B_{56}\}$	$\{B_{16}, B_{36}, B_{46}\}$	$\{F_{23}, F_{24}, F_{25}\}$	$B_{16} \oplus F_{23}$, $B_{36} \oplus F_{24}$, $B_{46} \oplus F_{25}$
(2,7)	$\{B_{17}, B_{37}, B_{47}, B_{57}\}$	$\{B_{67}\}$	$\{B_{17}, B_{57}\}$	$\{B_{67}, B_{37}, B_{47}\}$	$\{G_{23}, G_{24}, G_{25}\}$	$B_{67} \oplus G_{23}$, $B_{37} \oplus G_{24}$, $B_{47} \oplus G_{25}$
(3,6)	$\{C_{16}, C_{26}, C_{46}, C_{56}\}$	$\emptyset$	$\{C_{46}\}$	$\{C_{16}, C_{26}, C_{56}\}$	$\{F_{34}, F_{35}, F_{36}\}$	$C_{16} \oplus F_{34}$, $C_{26} \oplus F_{35}$, $C_{56} \oplus F_{36}$
(3,7)	$\{C_{17}, C_{27}, C_{47}, C_{57}\}$	$\{C_{67}\}$	$\{C_{47}, C_{57}\}$	$\{C_{67}, C_{17}, C_{27}\}$	$\{G_{34}, G_{35}, G_{36}\}$	$C_{67} \oplus G_{34}$, $C_{17} \oplus G_{35}$, $C_{27} \oplus G_{36}$
(4,6)	$\{D_{16}, D_{26}, D_{36}, D_{56}\}$	$\emptyset$	$\{D_{36}\}$	$\{D_{16}, D_{26}, D_{56}\}$	$\{F_{45}, F_{46}, F_{47}\}$	$D_{16} \oplus F_{45}$, $D_{26} \oplus F_{46}$, $D_{56} \oplus F_{47}$
(4,7)	$\{D_{17}, D_{27}, D_{37}, D_{57}\}$	$\{D_{67}\}$	$\{D_{17}, D_{37}\}$	$\{D_{67}, D_{27}, D_{57}\}$	$\{G_{45}, G_{46}, G_{47}\}$	$D_{67} \oplus G_{45}$, $D_{27} \oplus G_{46}$,$D_{57} \oplus G_{47}$
(5,6)	$\{E_{16}, E_{26}, E_{36}, E_{46}\}$	$\emptyset$	$\{E_{26}\}$	$\{E_{16}, E_{36}, E_{46}\}$	$\{F_{15}, F_{56}, F_{57}\}$	$E_{16} \oplus F_{15}$, $E_{36} \oplus F_{56}$, $E_{46} \oplus F_{57}$
(5,7)	$\{E_{17}, E_{27}, E_{37}, E_{47}\}$	$\{E_{67}\}$	$\{E_{27}, E_{37}\}$	$\{E_{67}, E_{17}, E_{47}\}$	$\{G_{15}, G_{56}, G_{57}\}$	$E_{67} \oplus G_{15}$, $E_{17} \oplus G_{56}$, $E_{47} \oplus G_{57}$

TABLE III: Sets $\Omega_{i,j}$, $\Lambda_{i,j}$, $\Delta_{i,j}$, $\mathcal{F}_{i,j}$, $\mathcal{H}_{i,j}$ and the multicasted messages in step 3 of the delivery phase for Example 1.

$|\Lambda_{i,j}| = \left\lfloor \frac{k^l-1}{2} \right\rfloor$ due to the approximately uniform partitioning. Accordingly,

$$|\Omega_{i,j}| + |\Lambda_{i,j}| = \left\lfloor \frac{K-1}{2} + \frac{k^h-2}{2} \right\rfloor, \text{ or} \tag{12}$$

$$|\Omega_{i,j}| + |\Lambda_{i,j}| = \left\lceil \frac{K-1}{2} + \frac{k^h-2}{2} \right\rceil. \tag{13}$$

One can observe that, when $k^h \geq 2$, in both cases $|\Omega_{i,j}| + |\Lambda_{i,j}| \geq \frac{K-1}{2}$. Hence, from now on we assume $k^h \geq 2$, and if $k^h = 1$, then in the delivery phase, the high-level file is considered as a low-level file, thus the achievable rate becomes $(K-1)/2$. One can also observe that $|\Lambda_{i,j}| \leq |\mathcal{H}_{i,j}|$, since $k^l \leq K$. Let $n_{i,j}$ be the required cardinality for the set $\Delta_{i,j}$ according to (10), i.e., $n_{i,j} \triangleq |\Omega_{i,j}| + |\Lambda_{i,j}| - |\mathcal{H}_{i,j}|$, then we can consider any subset of $\Omega_{i,j}$ with cardinality $n_{i,j}$ as $\Delta_{i,j}$ to construct $\mathcal{F}_{i,j}$. We note that all the sub-files in $\cup_{i\in\mathcal{K}^h, j\in\mathcal{K}^l} \Delta_{i,j}$ will be delivered in the fourth step of our scheme. Hence, $\Delta_{i,j}$ are chosen in order to minimize the number of transmitted messages in the last step. To clarify, in Example 1, if the server transmits $E_{26} \oplus B_{56}$ in the last step of the delivery phase, then this implies that $B_{56} \in \Delta_{2,6}$ and $E_{26} \in \Delta_{5,6}$. Therefore, we need to construct the multicast messages that will be transmitted in the last step while satisfying the constraint $\Delta_{i,j} = n_{i,j}$, $\forall\, i \in \mathcal{K}^h, j \in \mathcal{K}^l$. We will address this issue in Section IV. Once $\Delta_{i,j}$ is known, $\mathcal{F}_{i,j}$ can be constructed easily as illustrated in Table III, which also lists all the multicasted messages in this step.

4) Intra-high-level delivery with multicasting gain of two: In the last step, the server multicasts the messages

$$\mathcal{B} = \{C_{46} \oplus D_{36}, E_{26} \oplus B_{56}, A_{27} \oplus B_{17}, D_{17} \oplus A_{47} \\ C_{57} \oplus E_{37}, C_{47} \oplus D_{37}, B_{57} \oplus E_{27}\}, \tag{14}$$

each of which is destined for two high-level users. This step is completed with unicasting the sub-file A_{46}. In the next section, we will explain how the multicast messages in (14) are constructed.

IV. Smart set construction procedure

In this section, we will explain the procedure for constructing the set of multicast messages, $\mathcal{B}$. The main concern of this procedure is to satisfy the constraint $|\Delta_{i,j}| = n_{i,j}$[3] for all $i \in \mathcal{K}^h$ and $j \in \mathcal{K}^l$, while constructing set $\mathcal{B}$. In the construction procedure, we consider the two cases, where k^h is even and odd, separately.

A. Even number of high-level users

Before presenting the algorithm for the even number of high-level users, we will briefly explain how set partitioning can be used for the construction of multicast messages. Consider $\mathcal{K}^h = \{1,2,3,4,5,6\}$ and a partition of $\mathcal{K}^h$ into subsets of size two, e.g., $\mathcal{P}^{\mathcal{K}^h} = \{\{1,2\},\{3,4\},\{5,6\}\}$. Then, for a particular $j \in \mathcal{K}^l$ each $\{i,k\} \in \mathcal{P}^{\mathcal{K}^h}$ can be converted into the following multicast message: $W_{d_i,\{k,j\}} \oplus W_{d_k,\{i,j\}}$. For instance, for the specified partition set $\mathcal{P}^{\mathcal{K}^h}$, following multicast messages will be constructed: $W_{d_1,\{2,j\}} \oplus W_{d_2,\{1,j\}}$, $W_{d_3,\{4,j\}} \oplus W_{d_4,\{3,j\}}$ and $W_{d_5,\{6,j\}} \oplus W_{d_6,\{5,j\}}$. Note that, for a particular $j \in \mathcal{K}^l$, if $\mathcal{P}^{\mathcal{K}^h}$ is used to construct the multicast messages as illustrated above, then exactly one sub-file is added to the set $\Delta_{i,j}$ for each $i \in \mathcal{K}^h$. Hence, for a particular $j \in \mathcal{K}^l$, if we have n_j disjoint partitions $\mathcal{P}_1^{\mathcal{K}^h}, \ldots, \mathcal{P}_{n_j}^{\mathcal{K}^h}$ to construct multicast messages, then the constraint $|\Delta_{i,j}| = n_j$ is satisfied for all $i \in \mathcal{K}^h$. Therefore, we use $\mathcal{Q}^{n_j} = \cup_{i=1:n_j} \mathcal{P}_i^{\mathcal{K}^h}$ to construct multicast messages in Algorithm 1. We note that $\mathcal{Q}^n$ can be considered as a set of user pairings, where each user appears in exactly n pairings. Details on the construction of disjoint partition sets $\mathcal{P}_1^{\mathcal{K}^h}, \ldots, \mathcal{P}_n^{\mathcal{K}^h}$ are given in [12].

B. Odd number of high level users

In Section IV-A, we first constructed a set of node pairings $\mathcal{Q}^{n_j}$ for each $j \in \mathcal{K}^l$, and then generated the multicast messages using the node pairings. In the construction of $\mathcal{Q}^{n_j}$,

[3]Since the same partitioning is applied to all Λ_i's, $\forall i \in \mathcal{K}^h$, $n_{i,j} = n_j$ for all $i \in \mathcal{K}^h$.

Algorithm 1: Set construction for the case of even k^h

Input : $\mathcal{K}^h$, $\{n_j\}_{j\in\mathcal{K}^l}$
Output: $\{\Delta_{i,j}\}_{i\in\mathcal{K}^h,j\in\mathcal{K}^l}$, $\mathcal{B}$

$\Delta_{i,j} \leftarrow \{\}$ for all $i \in \mathcal{K}^h, j \in \mathcal{K}^l$;
$\mathcal{B} \leftarrow \{\}$;
for *all* $j \in \mathcal{K}^l$ **do**
  construct $\mathcal{Q}^{n_j}$, then;
  for *all* $\{i,k\} \in \mathcal{Q}^{n_j}$ **do**
    $\Delta_{i,j} \leftarrow \Delta_{i,j} \bigcup \{W_{d_i,\{k,j\}}\}$;
    $\Delta_{k,j} \leftarrow \Delta_{k,j} \bigcup \{W_{d_k,\{i,j\}}\}$;
    $\mathcal{B} \leftarrow \mathcal{B} \bigcup \{W_{d_i,\{k,j\}} \oplus W_{d_k,\{i,j\}}\}$;
  end
end

we use the partition sets $\mathcal{P}^{\mathcal{K}^h}$. However, when k^h is an odd number, it is not possible to partition $\mathcal{K}^h$ into subsets of cardinality two. Note that, if k^h is an odd number, k^l must be an even number, which means that for $k^l/2$ low-level users, n_j will be an odd number n_{odd}, and for the remaining low-level users, n_j will be an even number n_{even}. Let $\mathcal{K}^l_{odd}$ and $\mathcal{K}^l_{even}$ be the subset of low-level users with n_{odd} and n_{even}, respectively. Furthermore, let k^l_{odd} and k^l_{even} denote the cardinality of the sets $\mathcal{K}^l_{odd}$ and $\mathcal{K}^l_{even}$, respectively.

For the case of n_{even}, we introduce a new method, Algorithm 2, to construct $\mathcal{Q}^{n_{even}}$ for each $j \in \mathcal{K}^l$. We note that, in Algorithm 2 we use the notation $\mathcal{K}^l_{even}(ind)$ and $\mathcal{P}(ind)$ to denote the elements with index[4] ind in the given sets. We also want to remark that partition sets[5] used in Algorithm 2 are disjoint, i.e., $\mathcal{P}^{\mathcal{K}^h\setminus\{i\}} \cap \mathcal{P}^{\mathcal{K}^h\setminus\{k\}} = \emptyset$, where $i,k \in \mathcal{K}^h$, and $i \neq k$. Hence, one can easily observe that, each high-level user appears exactly in one pairing in $\mathcal{P}^{\mathcal{K}^h\setminus\{i\}}$, except i, and in $\mathcal{P}^{\mathcal{K}^h\setminus\{i\}} \bigcup \mathcal{P}^{\mathcal{K}^h\setminus\{k\}} \cup \{i,k\}$ each high-level user appears exactly in two pairings. Eventually, in $\mathcal{Q}^{n_{even}}$ each high-level user appears exactly in n_{even} pairings. Hence, as in Algorithm 1, $\mathcal{Q}^{n_{even}}$ can be used to construct sets $\Delta_{i,j}$ as well the set of multicast messages $\mathcal{B}$.

For each $j \in \mathcal{K}^l_{even}$ the same set of node pairings $\mathcal{Q}^{n_{even}}$ is used to construct the multicast messages $\mathcal{B}$; hence, the process is identical for each $j \in \mathcal{K}^l_{even}$. However, it is not possible to have a single $\mathcal{Q}^{n_{odd}}$ for each $j \in \mathcal{K}^l_{odd}$ to construct set $\mathcal{B}$. Nevertheless, we follow a similar procedure to construct the multicast messages for the low-level users in $\mathcal{K}^l_{odd}$. Our goal in this paper is to highlight the fundamental aspects of the CLCD scheme, while the complete algorithm for odd number of high-level users is provided in [12]. With the proposed set construction algorithms we are ensuring that in the third and fourth steps of the delivery phase all the sub-files are delivered with a multicasting gain of two[6], which explains the achievable delivery rate.

[4]Although we use index for the sets $\mathcal{K}^l_{even}$, $\mathcal{P}$, Algorithm 2 does not require a particular ordering for these sets

[5]Due to limited space we omitted the construction procedure of the partitions sets, for further details please refer to [12].

[6]When $|\cup_{i\in\mathcal{K}^h,j\in\mathcal{K}^l}\Delta_{i,j}|$ is odd, exactly one sub-file is unicasted, while the remaining sub-files achieve a multicasting gain of two, as in Example 1.

Algorithm 2: Construction of $\mathcal{Q}^{n_{even}}$ for odd k^h

Input : $\mathcal{K}^h, n_{even}$
Output: $\mathcal{Q}^{n_{even}}$

for $ind = 1 : n_{even}/2$ **do**
  $\{i,k\} \leftarrow \mathcal{P}^{\mathcal{K}^h\setminus\left\{\mathcal{K}^h(k^h)\right\}}(ind)$;
  $\mathcal{Q}^{n_{even}} \leftarrow \mathcal{Q}^{n_{even}} \bigcup \mathcal{P}^{\mathcal{K}^h\setminus\{i\}} \bigcup \mathcal{P}^{\mathcal{K}^h\setminus\{k\}} \cup \{i,k\}$;
end

V. CL(t, 1, 0) SCHEME

We can generalize the CL(t, 1) scheme to the CL(t, 1, 0) scheme, in which some of the files are not cached at all. We present this generalized scheme for the special case of $t = 2$.

A. CL(2, 1, 0) *Scheme*

In the CL(2, 1) scheme, we grouped the files according to their popularities. For a given cache size M, we can choose not to cache a certain number of least popular files, N_r, at all, in order to increase the number of high-level files N_h.

In the delivery phase, the users requesting files that are not cached are treated as virtual high-level users. To clarify, for a given demand realization $\mathbf{d}$, let $\mathbf{k}_{2-1-0} \triangleq [k^h, k^l, k^r]$, where k^r is the total number of users requesting one of the uncached files. If the users requesting these files are considered as virtual high-level users, the delivery phase becomes identical to the delivery phase of CL(2, 1) scheme for $\mathbf{k}_{2-1} = [k^h + k^r, k^l]$. Then, the sub-files of the uncached files are unicasted. We note that, in the first step of the delivery phase, the server does not need to send the multicast messages that are destined for only the virtual high-level users. Hence, the delivery rate for a demand realization $\mathbf{k}_{2-1-0}$ is given by

$$R(\mathbf{k_{2-1-0}}) = R(\mathbf{k}_{2-1}) + k^r - \frac{\binom{k^r}{3}}{\binom{K}{2}}. \tag{15}$$

Recall that the average delivery rate is

$$\bar{R} \triangleq \sum_{\mathbf{k}_{2-1-0}} R(\mathbf{k}_{2-1-0}) P_{N_h,N,N_r}(\mathbf{k}_{2-1-0}). \tag{16}$$

Thus, to minimize the average delivery rate we search for the optimal value of N_r, $N_r^\star$. We note that $N_r^{max} \geq N_r^\star \geq N_r^{min}$, where N_r^{max} and N_r^{min} are defined as the number of files that are not cached in order to cache all the remaining files at level $t = 2$ and $t = 1$, respectively. Hence, to find the optimal value of $N_r^\star$, average delivery rate $\bar{R}$ is calculated for each possible value of N_r within the interval $[N_r^{min}, N_r^{max}]$. We remark that, for each value of N_r, $P(\mathbf{k}_{2-1-0})$ must be calculated for each possible realization of $\mathbf{k}_{2-1-0}$. Hence, the optimization of the placement phase has a complexity of $\mathcal{O}(NK^2)$.

VI. NUMERICAL RESULTS

In this section, we compare the performance of the proposed CLCD scheme with that of the conventional centralized coded delivery scheme for two different content placement schemes. The first content placement scheme is called *naive memory sharing*, introduced in [1], in which all the files are cached identically according to a single parameter $t = MK/N$.

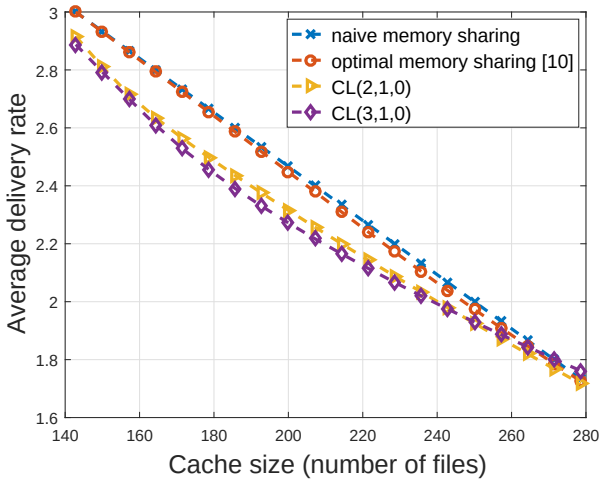

Fig. 1: $\gamma = 0.7$.

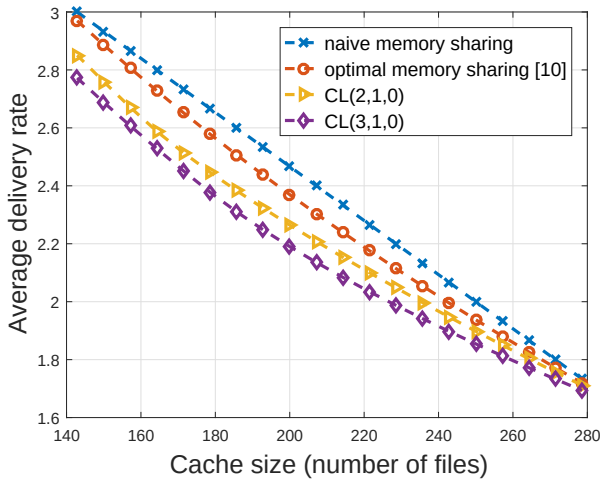

Fig. 2: $\gamma = 0.75$.

Note that, when the parameter t is not integer, we can apply memory-sharing [1] by dividing the files in the library into two disjoint fragments, and these fragments are cached according to parameters $\lfloor t \rfloor$ and $\lceil t \rceil$, respectively.

The second benchmark scheme is the *optimal memory sharing* scheme proposed in [10], which is shown to outperform other coded delivery techniques under non-uniform demand distributions. In this scheme, each file is divided into $K+1$ disjoint fragments, and the kth fragment, $1 < k$, is cached according to parameter $t = k-1$, while the first fragment is not cached. Thus, the overall system can be considered as a combination of $K+1$ sequential coded delivery phases with different multicasting gains, i.e., the kth delivery phase is executed with multicasting gain of $k+1$. The sizes of the fragments are obtained via a linear program.

In general, the popularity of video files for on-demand streaming applications approximately fits a Zipf distribution with parameter $1 > \gamma > 0.65$ [4], [5]. Hence, in our simulations we consider $\gamma = 0.7$ and $\gamma = 0.75$. In realistic scenarios, number of files in the video library is considered to be on the order of 10^4. However, due to the complexity of the optimal memory sharing scheme of [10], we assume $N = 1000$ and $K = 7$. In the simulations, the cache size M varies from 140 to 280, which corresponds to $1 < t < 2$. The delivery rates of the naive memory sharing, the optimal memory sharing scheme, $\mathrm{CL}(2,1,0)$ and $\mathrm{CL}(3,1,0)$[7] schemes are illustrated in Fig. 1 and Fig. 2, for $\gamma = 0.7$ and $\gamma = 0.75$, respectively. We observe that the proposed CLCD scheme reduces the average delivery rate especially for moderate cache capacities. While the gain from the $\mathrm{CL}(2,1,0)$ scheme reduces as the Zipf parameter increases, the gain provided by $\mathrm{CL}(3,1,0)$ increases.

VII. Conclusions

We introduced a novel centralized coded delivery scheme with uncoded caching for non-uniform demand distributions. Due to space limitations, we presented a special case of our scheme, called $\mathrm{CL}(2,1,0)$, which divides the library into three groups, and uses different levels of caching for the first two groups, while the last group of the least popular files are not cached at all. The delivery rate is presented in closed form as a function of the number of users requesting files from each group. Numerical simulations show that the proposed scheme can provide up to 10% reduction in the average delivery rate.

[7]The $\mathrm{CL}(3,1,0)$ scheme can be constructed similarly to $\mathrm{CL}(2,1,0)$; however, the details are skipped due to page limitation.

References

[1] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, May 2014.

[2] ——, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, Aug 2015.

[3] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," in *2017 IEEE Int. Symp. on Inf. Theory (ISIT)*, June 2017, pp. 1613–1617.

[4] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *2008 16th Int. Workshop on Quality of Service*, June 2008.

[5] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, Oct 2009.

[6] J. Hachem, N. Karamchandani, and S. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, May 2017.

[7] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, Feb 2017.

[8] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, June 2017.

[9] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," in *Inf. Theory and Appl. Works.*, Feb 2015.

[10] S. Jin, Y. Cui, H. Liu, and G. Caire, "Structural properties of uncoded placement optimization for coded delivery," *CoRR*, vol. abs/1707.07146, 2017.

[11] A. M. Daniel and W. Yu, "Optimization of heterogeneous coded caching," *CoRR*, vol. abs/1708.04322, 2017.

[12] E. Ozfatura and D. Gündüz, "Uncoded caching and cross-level coded delivery for non-uniform file popularity," 2018, in preparation.